

# Competitive-Ratio Approximation Schemes for Minimizing the Makespan in the Online-List Model

Nicole Megow\*

Andreas Wiese†

## Abstract

We consider online scheduling on multiple machines for jobs arriving one-by-one with the objective of minimizing the makespan. For any number of identical parallel or uniformly related machines, we provide a competitive-ratio approximation scheme that computes an online algorithm whose competitive ratio is arbitrarily close to the best possible competitive ratio. We also determine this value up to any desired accuracy. This is the first application of competitive-ratio approximation schemes in the online-list model. The result proves the applicability of the concept in different online models. We expect that it fosters further research on other online problems.

## 1 Introduction

Online scheduling problems have been studied extensively for more than two decades [22, 24]. One of the most extensively investigated problems among them is the makespan minimization problem with jobs arriving one-by-one: We are given  $m$  identical parallel machines, and we assume throughout the paper that  $m$  is an arbitrary but fixed constant. The set of jobs  $J = \{1, 2, \dots\}$  with integral processing times  $p_j \geq 1$  ( $j \in J$ ) is presented to the online algorithm one after the other. Once a job is present, it must be assigned without splitting, immediately, and irrevocably to a machine before the next job is revealed. This model for revealing online information one-by-one is called the *online-list* model [22]. The goal is to minimize the makespan, that is, the last completion time of all currently present jobs. Using the standard three-field notation [17], we denote the problem as the online-list variant of  $Pm||C_{\max}$ . We also consider the more general model of uniformly related machines  $Qm||C_{\max}$ , where each machine  $i \in \{1, \dots, m\}$  is given a speed  $s_i$  and the execution time of job  $j$  on machine  $i$  is  $p_j/s_i$ .

The performance of online algorithms is typically assessed by *competitive analysis* [20, 25] which determines the worst-case performance compared to an optimal offline algorithm. Let an instance  $I$  be defined by a set of jobs  $J$  with processing times  $p_j$  ( $j \in J$ ), and  $m$ , the number of available machines. And let  $\mathcal{I}_m$  be the set of instances with  $m$  machines. We call an online algorithm  $\rho(m)$ -competitive if, for any problem instance  $I \in \mathcal{I}_m$ , it achieves a solution with cost  $\text{ALG}(I) \leq \rho(m) \cdot \text{OPT}(I)$ , where  $\text{ALG}(I)$  and  $\text{OPT}(I)$  denote the solution value of the online and an optimal offline algorithm, respectively, for the same instance  $I$ . The *competitive ratio*  $\rho_{\text{ALG}}(m)$  of  $\text{ALG}$  is the infimum over all  $\rho$  such that  $\text{ALG}$  is  $\rho$ -competitive. The minimum competitive ratio  $\rho^*(m)$  achievable by any online algorithm for instances in  $\mathcal{I}_m$  is called *optimal for  $m$  machines*. The optimal competitive ratio over all number of machines is  $\rho^* := \max_{m \in \mathbb{N}} \rho^*(m)$ .

Only recently, the concept of competitive-ratio approximation schemes was introduced in [18]. Such an approximation scheme is a procedure that computes a nearly optimal online algorithm and at the same time provides a nearly exact estimate of the optimal competitive ratio. The general definition (without distinguishing by  $m$ ) is as follows.

---

\*Department of Mathematics, Technische Universität Berlin, Germany. Email: [nmegow@math.tu-berlin.de](mailto:nmegow@math.tu-berlin.de). Supported by the German Science Foundation (DFG) under contract ME 3825/1.

†Max-Planck-Institut für Informatik, Saarbrücken, Germany. Email: [wiese@mpi-inf.mpg.de](mailto:wiese@mpi-inf.mpg.de).

**Definition 1.** A competitive-ratio approximation scheme *computes for a given  $\varepsilon > 0$  an online algorithm ALG with a competitive ratio  $\rho_{\text{ALG}} \leq (1 + \varepsilon)\rho^*$ . Moreover, it determines a value  $\rho'$  such that  $\rho' \leq \rho^* \leq (1 + \varepsilon)\rho'$ .*

In this paper we provide a competitive-ratio approximation scheme for the online-list variant of makespan minimization on identical parallel and uniformly related machines for any number of machines. This is the first competitive-ratio approximation scheme for a problem in the *online-list* model in contrast to previous work in the so-called *online-time* model [22]. In the latter model jobs are revealed to the algorithm online over time at their individual release date. Regarding the decision making process, an online algorithm has more freedom in this model, as it is allowed to postpone decisions or even revoke them as long as the jobs have not been executed.

## 1.1 Related Work

The online-list makespan minimization problem has been studied extensively—mainly, on identical parallel machines. The classical list scheduling algorithm with competitive ratio  $2 - 1/m$  [16] is optimal for  $m \in 2, 3$  [13]. For  $m \geq 4$  better algorithms have been proposed and improved general lower bounds were shown in a series of works [2–4, 6, 13–15, 19, 23]. The currently best known bounds on the optimal competitive ratio  $\rho^*(m)$  for some particular values of  $m$  are  $\rho^*(4) \in [1.732, 1.733]$ ,  $\rho^*(5) \in [1.770, 1.746]$ ,  $\rho^*(6) \in [1.8, 1.773]$ ,  $\dots$ ,  $\rho^* \in [1.88, 1.9201]$ .

On uniformly related machines the gap is much larger. The lower bound on the competitive ratio for an arbitrary number of machines is 2.564 [11], while the currently best known upper bound is 5.828 [5]. Interestingly, the special case of two related machines is completely solved, meaning that the exact competitive ratio known [9] and even stronger, the exact ratio for any pair of speeds is known [12, 26].

We also remark that the preemptive variant of the identical machine problem is completely solved and the optimal competitive ratio for any number of machines [7] is known. For uniformly related machines, an optimal online algorithm is known for any number of machines and any combination of speeds [10]. Interestingly, from their linear programming based approach it is not clear how to derive the actual value of the competitive ratio except for  $m \in \{3, 4\}$ .

Competitive-ratio approximation schemes were introduced by Günther et al. in [18]. They focussed on scheduling problems in the online-time model and provide such schemes for various scheduling problems  $\text{Pm} | r_j, (pmtn) | \sum w_j C_j$ ,  $\text{Qm} | r_j, (pmtn) | \sum w_j C_j$  (assuming a constant range of machine speeds without preemption), and  $\text{Rm} | r_j, pmtn | \sum w_j C_j$ . They also consider minimizing the makespan,  $C_{\max}$ , and  $\sum_{j \in J} w_j f(C_j)$ , where  $f$  is an arbitrary monomial function with fixed exponent. Subsequently, Kurpisz et al. [21] showed how to construct competitive-ratio approximation schemes for  $\text{Rm} | r_j | C_{\max}$ , makespan minimization in a job shop problem, and scheduling with delivery times—again, all in the online-time model.

We are not aware of any publication of similar results for the online-list model<sup>1</sup>. Notice that the results in [10] are conceptually strongly related. The main difference is that our approximation scheme provides the algorithmic means to compute the actual value of the optimal competitive ratio (up to some error), whereas this remains open for the algorithm in [10] even though it is provably optimal.

## 1.2 Our Results

In this paper we provide competitive-ratio approximation schemes for the online-list variants of makespan minimization on identical parallel and uniformly related machines for any number of machines, that is,  $\text{Pm} | C_{\max}$  and  $\text{Qm} | C_{\max}$ . More precisely, given  $\varepsilon > 0$  and an  $m \in \mathbb{N}$ , we provide an online algorithm  $\text{ALG}(m)$  with a competitive ratio  $\rho_{\text{ALG}}(m) \leq (1 + \varepsilon)\rho^*(m)$ . Moreover, it determines a value  $\rho'$  such that  $\rho' \leq \rho^*(m) \leq (1 + \varepsilon)\rho'$ .

On a high level, we use a similar approach as in [18]. We first simplify and structure the input without changing the instance too much, and then we reduce the complexity of possible

<sup>1</sup>However, as of writing this we got contacted by another group of researchers [8] who obtained similar results as ours, independently.

online algorithms by interpreting them as an algorithm map that bases its decisions only on the currently unfinished jobs and the schedule history. The key insight is that a very limited (constant) part of the schedule history is sufficient to take decisions which are close to optimal. In contrast to the previous work in the online-time model, our amount of history that has to be considered depends on the size of the currently largest revealed job instead of the time at which the last jobs were released.

The main purpose of this paper is to provide a proof of concept for competitive-ratio approximation schemes and to show that it is also applicable to online problems in the online-list model. The actual gaps between upper and lower bounds on the optimal competitive ratios are rather small on identical parallel machines, but this is one of the most classical online-list problems. Since many online problems follow the online-list paradigm, we hope that this work fosters further research on competitive-ratio approximation schemes.

**Outline of the paper.** We first consider identical parallel machines in Section 2. Then we argue on how to extend this technique to uniformly related machines in Section 3. We conclude with open questions and further research potential.

## 2 Identical Parallel Machines

We give a competitive-ratio approximation scheme for the online-list variant of  $Pm||C_{\max}$ . We first give some transformations that simplify the input and reduce the structural complexity of online schedules. Then we use the an abstract view on online algorithms to reduce complexity further and to describe the approximation scheme.

### 2.1 Restrictions at $1 + \varepsilon$ loss

We will use the terminology that *at  $1 + \varepsilon$  loss we can restrict* to instances or schedules with certain properties. This means that we lose at most a factor  $1 + \varepsilon$  in the objective value, as  $\varepsilon \rightarrow 0$ , by limiting our attention to those instances. We bound several relevant parameters by constants. If not stated differently, any mentioned constant depends only on  $\varepsilon$  and  $m$ .

In the online-list model we refer to an iteration for each job arrival. We will slightly abuse notation and refer to the iteration in which job  $j$  is revealed as iteration  $j$ . For an online algorithm  $A$ , an instance  $I$ , and an iteration  $j$ , denote by  $A_j(I)$  the makespan of the schedule obtained after iteration  $j$  when processing instance  $I$  by algorithm  $A$ . Furthermore, we define  $p(J) := \sum_{j \in J} p_j$ .

The first observation has been made already in other contexts, e.g., in [1] for minimizing  $\sum w_j C_j$ .

**Proposition 2.** *At  $1 + \varepsilon$  loss we can restrict to instances where all  $p_j$  are powers of  $1 + \varepsilon$ .*

In order to simplify the construction of our algorithms, we can actually restrict to instances with a very simple and special structure.

**Lemma 3.** *At  $1 + \varepsilon$  loss we can restrict to instances where for each  $k \in \mathbb{N}$  there are at most  $\frac{m}{\varepsilon^3}$  jobs  $j$  with  $p_j = (1 + \varepsilon)^k$ .*

*Proof.* Suppose that we have an online algorithm  $A'$  which achieves a competitive ratio of  $\rho_{A'}(m)$  on instances with at most  $\frac{m}{\varepsilon^3}$  jobs, each with processing time  $p_j = (1 + \varepsilon)^k$  for each  $k \in \mathbb{N}$ . Based on  $A'$  we construct an online algorithm  $A$  for arbitrary instances (assuming that processing times are powers of  $1 + \varepsilon$ ) with a competitive ratio of  $\rho_A(m) \leq (1 + O(\varepsilon))\rho_{A'}(m)$ .

Suppose that we are given an (arbitrary) instance  $I$  where all  $p_j$  are powers of  $1 + \varepsilon$ . We construct an instance  $I'$  which we present to  $A'$ . Based on the schedule  $A'(I')$  we construct the schedule  $A(I)$ . As long as for each  $k \in \mathbb{N}$  at most  $\frac{m}{\varepsilon^3}$  jobs  $j$  with  $p_j = (1 + \varepsilon)^k$  are released, the instances  $I$  and  $I'$  are identical and we define  $A(I)$  to be identical to  $A'(I')$ . Now suppose that in some iteration  $j$  a job  $j$  with  $p_j = (1 + \varepsilon)^k$  is revealed after there where released already  $\frac{m}{\varepsilon^3}$  other jobs with the same processing time. Let  $I'_j$  denote the instance up to job  $j$ . Observe that  $\text{OPT}(I'_j) \geq \frac{1}{\varepsilon^3}(1 + \varepsilon)^k$ . Let  $p := (1 + \varepsilon)^{\lceil \log_{1+\varepsilon} \varepsilon^2 \cdot \text{OPT}(I'_j) \rceil} \geq \varepsilon^2 \cdot \text{OPT}(I'_j)$ .

We observe that so far at most  $\frac{m}{\epsilon^2} < \frac{m}{\epsilon^3}$  jobs of size  $p$  have been released since otherwise  $\text{OPT}(I'_j) \geq \frac{1}{m}(\frac{m}{\epsilon^2} + 1) \cdot p > \frac{1}{m} \cdot \frac{m}{\epsilon^2} \cdot \epsilon^2 \text{OPT}(I'_j) = \text{OPT}(I'_j)$ . Instead of  $j$ , in instance  $I'$  we release a new job  $j'$  with  $p_{j'} = p$ . Suppose that algorithm  $A'$  assigns  $j'$  on machine  $i$ . Then, algorithm  $A$  assigns the next upcoming jobs  $p_{j''}$  with  $p_{j''} \leq p_j$  to  $i$ , as long as their total processing time is bounded by  $p$ . More precisely, we define  $j_{\max}$  to be the maximum value such that for the set  $J := \{j'' \in I \mid j \leq j'' \leq j_{\max} \wedge p_{j''} \leq p_j\}$  it holds that  $p(J) \leq p$ . We define that algorithm  $A$  assigns all jobs in  $J$  to machine  $i$ . We call  $j'$  a *container job*. We say that after iteration  $j_{\max}$  the container job  $j'$  is *full*. Intuitively this means that we do not add any further jobs to  $j'$ . At each iteration  $j''$  we say that the jobs in  $J \cap J_{j''}$  are in the container  $j'$ . Observe that  $p_{j'} \leq p_j = (1 + \epsilon)^k \leq \epsilon^3 \cdot \text{OPT}(I'_j) \leq \epsilon \cdot p$  for all  $j' \in J$ .

By construction, we observe that for each  $k \in \mathbb{N}$  there is at most one container job of size  $(1 + \epsilon)^k$  which is less than a  $(1 - \epsilon)$ -fraction full. In particular, if we create a new container job of size  $p := (1 + \epsilon)^{\lceil \log_{1+\epsilon} \epsilon^2 \cdot \text{OPT}(I'_j) \rceil}$  then up to iteration  $j$  strictly less than  $\frac{m}{\epsilon^3}$  jobs (container jobs and normal jobs!) of size  $p$  have been released since otherwise  $\text{OPT}(I'_j) > \frac{1}{m}(\frac{m}{\epsilon^3} \cdot p) \geq \frac{1}{m} \cdot \frac{m}{\epsilon^3} \cdot \epsilon^2 \text{OPT}(I'_j) \geq \text{OPT}(I'_j)$ . According to the above definition, it can happen that we open a new container job while an old container with smaller size is not yet to a  $(1 - \epsilon)$ -fraction full. In this case we close the smaller old container and do not add any further jobs to it.

To prove the competitive ratio of  $A'$  we need to show that  $\frac{A(I_j)}{\text{OPT}(I_j)} \leq (1 + O(\epsilon)) \frac{A'(I'_j)}{\text{OPT}(I'_j)} \leq (1 + O(\epsilon)) \rho_{A'}(m)$ . By construction we have that  $A(I_j) \leq A'(I'_j)$  since  $A$  and  $A'$  assign the jobs in  $J(I_j) \cap J(I'_j)$  to the same machines and on each machine  $i$  the total processing time of the jobs in  $J(I_j) \setminus J(I'_j)$  is bounded from above by the total processing time of the jobs in  $J(I'_j) \setminus J(I_j)$  (the container jobs) on this machine. It remains to show that  $\text{OPT}(I'_j) \leq (1 + O(\epsilon)) \text{OPT}(I_j)$ . Based on  $\text{OPT}(I_j)$  we construct a schedule  $S$  for  $I'_j$  whose makespan is bounded by  $(1 + O(\epsilon)) \text{OPT}(I_j)$ . In  $S$ , we assign all jobs in  $J(I_j) \cap J(I'_j)$  to the same machine as in  $\text{OPT}(I_j)$ . Then, we assign the jobs in  $J(I'_j) \setminus J(I_j)$  (the container jobs) greedily. If after the greedy assignment the global makespan does not change, then  $\text{OPT}(I'_j) \leq \text{OPT}(I_j)$ .

Now suppose that after the greedy assignment the global makespan increases. Then the load of any two machines can differ by at most  $\tilde{p}$  which denotes the maximum processing time of a container job in  $I'_j$ . Note that  $\tilde{p} \leq (1 + \epsilon)^{\lceil \log_{1+\epsilon} \epsilon^2 \cdot \text{OPT}(I'_j) \rceil}$  and observe that the makespan of  $S$  is upper-bounded by  $\frac{1}{m} \cdot p(I'_j) + \tilde{p}$ . Since for each  $k \in \mathbb{N}$  there is at most one container job of size  $(1 + \epsilon)^k$  which is less than a  $(1 - \epsilon)$ -fraction full we further conclude that

$$\begin{aligned}
\text{OPT}(I'_j) &\leq \frac{1}{m} \cdot p(I'_j) + \tilde{p} \\
&\leq \underbrace{\frac{1}{m}(1 + O(\epsilon))p(I_j)}_{\text{normal jobs and } (1-\epsilon)\text{-full container jobs}} + \tilde{p} + \underbrace{\sum_{1 \leq k' \leq \lceil \log_{1+\epsilon} \epsilon^2 \cdot \text{OPT}(I'_j) \rceil} (1 + \epsilon)^{k'}}_{\text{less than } (1-\epsilon)\text{-full container jobs}} \\
&\leq \frac{1}{m}(1 + O(\epsilon))p(I_j) + 2(1 + \epsilon)^{\lceil \log_{1+\epsilon} \epsilon^2 \cdot \text{OPT}(I'_j) \rceil} + \sum_{1 \leq k' \leq \log_{1+\epsilon} \epsilon^2 \cdot \text{OPT}(I'_j)} (1 + \epsilon)^{k'} \\
&\leq \frac{1}{m}(1 + O(\epsilon))p(I_j) + 2(1 + \epsilon)^{\lceil \log_{1+\epsilon} \epsilon^2 \cdot \text{OPT}(I'_j) \rceil} + \frac{1}{\epsilon}(1 + \epsilon)^{1 + \log_{1+\epsilon}(\epsilon^2 \cdot \text{OPT}(I'_j))} \\
&\leq \frac{1}{m}(1 + O(\epsilon))p(I_j) + 2(1 + \epsilon)\epsilon^2 \cdot \text{OPT}(I'_j) + \frac{1 + \epsilon}{\epsilon}(\epsilon^2 \cdot \text{OPT}(I'_j)) \\
&\leq (1 + O(\epsilon))\text{OPT}(I_j) + O(\epsilon) \cdot \text{OPT}(I'_j)
\end{aligned}$$

which implies that  $\text{OPT}(I'_j) \leq (1 + O(\epsilon))\text{OPT}(I_j)$ .  $\square$

## 2.2 Online Algorithms and Algorithm Maps

As in [18] we use an abstract characterization of an online algorithm and interpret it as a map. The map gets as input the so far computed schedule and the size of the next released job  $j$ . Based on these data, it decides to which machine it assigns  $j$ .

To this end, we define a configuration  $C$  as follows.

**Definition 4.** A configuration  $C$  is the combination of

- a set  $J(C)$  of previously released jobs, including their order,
- a map  $\chi_C : J(C) \rightarrow \{1, \dots, m\}$  which defines the assignment of the jobs in  $J(C)$  to the machines,
- the processing time  $p_{j^*}$  of the newly released but not yet assigned job.

We will write only  $J$  or  $\chi$  (rather than  $J(C)$  or  $\chi_C$ ) when  $C$  is clear from the context. Let  $\mathcal{C}$  denote the (infinite) set of configurations. We say that a configuration  $C$  is in *phase*  $k$  if  $\max_{j \in J(C) \cup \{j^*\}} p_j = (1 + \varepsilon)^k$ . Let  $C$  be a configuration in phase  $k$ . We call a job  $j \in J(C)$  *relevant* if  $p_j \geq (1 + \varepsilon)^{k-s}$  where  $s \in \mathbb{N}$  is the smallest integer such that  $s \geq \log_{1+\varepsilon} \frac{m(1+\varepsilon)}{\varepsilon^5}$  (note that  $s$  depends only on  $\varepsilon$  and  $m$  and is independent of  $C$ ). Denote by  $J_R(C) \subseteq J(C) \cup \{j^*\}$  all *relevant* jobs for a configuration  $C$ . It will turn out later that at  $1 + \varepsilon$  loss we can neglect the jobs which are not relevant, which we will call *irrelevant*. Define by  $\text{MS}(C) := \max_i p(\chi_C^{-1}(i))$  the *makespan* of  $C$ . Also, we define  $\text{OPT}(C) := \text{OPT}(J(C))$  (note here that  $J(C)$  does not include the newly released job  $j^*$ ).

We interpret an online algorithm for our problem on  $m$  machines as a map  $f : \mathcal{C} \rightarrow \{1, \dots, m\}$ : Given a configuration  $C$  with a newly released job  $j^*$ , the algorithm map  $f$  assigns  $j^*$  to the machine  $f(C)$ . Like for online algorithms, we denote by  $\rho_f(m)$  the competitive ratio obtained by the map  $f$ .

**Proposition 5.** For each online algorithm  $A$  for the problem on  $m$  machines there is an algorithm map  $f$  such that  $\rho_f(m) = \rho_A(m)$ .

**Definition 6.** Let  $C, C'$  be two configurations which are in phases  $k$  and  $k'$ , respectively. They are equivalent if there is a bijection  $\sigma : J_R(C) \rightarrow J_R(C')$  such that

- $p_{\sigma(j)} = (1 + \varepsilon)^{k'-k} \cdot p_j$ ,
- $\chi_{C'}(j) = \chi_C(\sigma(j))$  for all  $j \in J_R(C) \setminus \{j^*\}$ , and
- $\sigma(j^*) = j'^*$  if  $j^* \in J_R(C)$ , where  $j'^*$  denotes the newly released job in  $C'$ .

**Proposition 7.** There are only constantly many equivalence classes of configurations.

In Definition 6 we neglect the jobs which are not relevant. This is justified by the following lemma.

**Lemma 8.** Let  $C$  be a configuration for a phase  $k$ . Then  $p(J(C) \setminus J_R(C)) \leq \varepsilon \cdot (1 + \varepsilon)^k \leq \varepsilon \cdot \text{OPT}(C) \leq \varepsilon \cdot \text{MS}(C)$ .

*Proof.* Recall that by Lemma 3 we assumed that for each  $k' \in \mathbb{N}$  there are at most  $\frac{m}{\varepsilon^3}$  jobs  $j$  with  $p_j = (1 + \varepsilon)^{k'}$ . Hence, the total processing time of irrelevant jobs in  $J(C)$  is bounded by

$$\begin{aligned} p(J(C) \setminus J_R(C)) &\leq \sum_{1 \leq k' \leq k-s} \frac{m}{\varepsilon^3} (1 + \varepsilon)^{k'} \leq \frac{m}{\varepsilon^3} \cdot \frac{(1 + \varepsilon)^{k-s+1}}{\varepsilon} = (1 + \varepsilon)^k \frac{m(1 + \varepsilon)^{1-s}}{\varepsilon^4} \\ &\leq \varepsilon (1 + \varepsilon)^k. \end{aligned}$$

where the last inequality follows since  $\frac{m(1+\varepsilon)^{1-s}}{\varepsilon^4} \leq \varepsilon$  by definition of  $s$ . Since  $C$  is a configuration in phase  $k$ , and thus, by definition a job  $j$  with  $p_j = (1 + \varepsilon)^k$  must have been released. It follows that  $\varepsilon \cdot (1 + \varepsilon)^k \leq \varepsilon \cdot \text{OPT}(C) \leq \varepsilon \cdot \text{MS}(C)$ .  $\square$

In particular, two equivalent configurations have almost the same makespan and their respective jobs have almost the same optimal makespan.

**Lemma 9.** Let  $C, C'$  be two equivalent configurations for phases  $k$  and  $k'$ , respectively. Then  $\text{MS}(C') \leq (1 + O(\varepsilon))(1 + \varepsilon)^{k'-k} \cdot \text{MS}(C)$  and  $\text{OPT}(C') \leq (1 + O(\varepsilon))(1 + \varepsilon)^{k'-k} \cdot \text{OPT}(C)$ .

*Proof.* Follows from Definition 6 and Lemma 8.  $\square$

**Lemma 10.** *At  $1 + \varepsilon$  loss we can restrict to instances  $I$  such that for each iteration  $j$  we have that  $p_j \geq \max_{j' < j} p_{j'} \cdot (1 + \varepsilon)^{-s}$ .*

*Proof.* When in some iteration  $j$  a job  $j$  is released with  $p_j < \max_{j' < j} p_{j'} \cdot (1 + \varepsilon)^{-s}$  we assign  $j$  to some arbitrary machine. By Lemma 8 the total processing time of such jobs, released up to some iteration  $j'$ , is bounded by  $\epsilon \cdot \text{OPT}_{j'}$ .  $\square$

**Lemma 11.** *At  $1 + \varepsilon$  loss we can restrict to algorithm maps  $f$  such that  $f(C) = f(C')$  for any two equivalent configurations  $C$  and  $C'$ .*

*Proof.* Let  $f$  be an algorithm map without this property with a competitive ratio of  $\rho_f(m)$  on  $m$  machines. Based on  $f$  we construct a new algorithm map  $g$  with the claimed property such that  $\rho_g(m) \leq (1 + \varepsilon)\rho_f(m)$ .

We call a configuration  $C$  *realistic* for  $f$ , if there is an instance  $I$  such that  $f$  ends in configuration  $C$  when processing  $I$ . For each equivalence class  $\mathcal{C}'$  of configurations containing at least one realistic configuration, we pick a realistic representant  $C \in \mathcal{C}'$ . We say that  $C$  *represents*  $\mathcal{C}'$ . For all configurations  $C' \in \mathcal{C}'$  equivalent to  $C$  we define  $g$  such that  $g(C') = f(C)$ .

We claim that  $g$  is always in a configuration  $C$  such that there is a configuration  $C'$  with  $C \sim C'$  such that  $C'$  is realistic for  $f$ . We prove the claim by induction over the iterations. We start with the base case of zero previous iterations. Let  $C$  be a configuration which is realistic for  $g$ . Since so far no jobs have been scheduled  $C$  is also realistic for  $f$ . Now suppose that the claim is true when the first  $\ell$  jobs have been released. Suppose that after  $\ell$  jobs have been released  $g$  is in a configuration  $C$  for a phase  $k$  such that some configuration  $C'$  for phase  $k'$  with  $C \sim C'$  is realistic for  $f$ . Assume w.l.o.g. that  $C'$  represents its equivalence class. Assume that in  $C$  a job  $j^*$  with processing time  $p_{j^*}$  is released and denote by  $j'^*$  the newly released job in  $C'$ .

By construction,  $g$  assigns  $j$  to the machine  $g(C) = f(C')$ . Then a new (relevant) job  $\bar{j}$  is released which yields the configuration  $\bar{C}$ . Denote by  $\bar{C}'$  the configuration which results from  $C'$  after assigning job  $j'$  to machine  $f(C')$  and the release of a new job  $\bar{j}'$  with  $p_{\bar{j}'} = (1 + \varepsilon)^{k' - k} p_{\bar{j}}$ . Since  $\bar{j}$  is relevant for  $\bar{C}$ , it follows that  $\bar{j}'$  is relevant for  $\bar{C}'$ . Since  $C \sim C'$  and  $p_{\bar{j}'} = (1 + \varepsilon)^{k' - k} p_{\bar{j}}$  we conclude that  $\bar{C} \sim \bar{C}'$ .  $\square$

The decision of an algorithm map (with all above simplifications) for a configuration  $C$  depends only on the equivalence class of  $C$ . Since there are only constantly many equivalence classes for configurations (see Proposition 7) and for each configuration there are only  $m$  possible decisions, there are only constantly many algorithm maps. Hence, we can enumerate them all. With the procedure given by the following lemma we estimate its competitive ratio. Finally, we output the map with the minimum estimated competitive ratio.

**Lemma 12.** *Let  $f$  be an algorithm map for  $m$  machines. There exists an algorithm which computes a value  $\bar{\rho}$  with  $\rho_f(m) \leq \bar{\rho} \leq (1 + \varepsilon)\rho_f(m)$ .*

*Proof.* In order to determine  $\rho_f(m)$  it is sufficient to know all possible realistic configurations for  $f$ . By Lemma 3, the realistic configuration  $C$  with the worst competitive ratio determines  $\rho_f(m)$  up to an error of  $1 + \varepsilon$ .  $\square$

Combining all statements gives our main theorem.

**Theorem 13.** *There is a competitive-ratio approximation scheme for the online-list variant of the problem  $\text{Pm}||C_{\max}$  for any number of machines  $m$ .*

### 3 Uniformly Related Machines

With small additional instance transformations we can apply a similar construction in the setting of uniformly related machines. By scaling processing times and machine speeds, we can assume w.l.o.g. that the slowest machine has unit speed. Let  $s_{\max}$  denote the speed of the fastest machine in a given instance.

**Proposition 14.** *At  $1 + \varepsilon$  loss we can assume that the speed of each machine is a power of  $1 + \varepsilon$ .*

**Lemma 15.** *At  $1 + \varepsilon$  loss, we can restrict to instances in which  $s_{\max}$  is bounded by  $m/\varepsilon$ .*

*Proof.* Take a given schedule with makespan  $MS$  on related machines with speed values  $s_1, \dots, s_{\max}$ . For each machine whose speed is at most  $\frac{\varepsilon}{m} \cdot s_{\max}$ , we take the jobs assigned to it and add them to the fastest machine. The moved processing volume increases the total processing volume on the fast machine by at most  $\frac{\varepsilon}{m} \cdot MS$ . Thus, we can simply ignore machines whose speed is at most  $\frac{\varepsilon}{m} \cdot s_{\max}$ . The remaining machines have speeds in the range of  $[\frac{\varepsilon}{m} \cdot s_{\max}, s_{\max}]$ . Since we assume that the slowest machine has unit speed, after rounding the speeds we have that  $s_{\max} \leq m/\varepsilon$ .  $\square$

Hence, for each value  $m$  there are only finitely many speed vectors  $s_1, \dots, s_m$ . For each of these speed vectors, we can bound the number of jobs of the same size similarly to Lemma 3 with an additional dependence on  $s_{\max} \leq m/\varepsilon$ . This allows us to define configurations, algorithm maps, and equivalence relations similarly as in the previous section.

**Theorem 16.** *There is a competitive-ratio approximation scheme for the online-list variant of the problem  $Qm||C_{\max}$  for any number of machines  $m$ .*

## 4 Conclusion and Further Research

We provide competitive-ratio approximation schemes for the makespan minimization problem when jobs arrive online over a list. This proves that the concept of competitive-ratio approximation schemes is not limited to online (scheduling) problems in the online-time model.

The approximation schemes presented in this paper, compute a nearly optimal solution for any number of machines. On the theoretical side, it would be interesting to give a general approximation of the optimal competitive ratio over all possible numbers of machines  $m$ . This requires a better understanding of how  $\rho^*(m)$  behaves as a function of  $m$ . We know that it is bounded from above (for every  $m$ ). It seems intuitively (and suggested by known bounds) true that it is increasing in  $m$ .

Our approximation schemes do not only determine nearly best possible online algorithms, they also provide the algorithmic tools to compute the value of the optimal competitive ratio up to any desired accuracy. This is interesting because it contrasts the common approach to derive upper and lower bounds on the (optimal) competitive ratio manually. In particular, our theory proves that a computer may execute the algorithm to compute the desired bounds. However, the drawback of our presented construction is its computational complexity. To reduce the gaps between the currently best known upper and lower bounds, we would have to chose a quite small accuracy parameter  $\varepsilon$  which leads to a hopeless running time. We believe that a more careful design of the necessary input simplification and algorithm structuring might lead to approximation schemes that can compute explicitly the value of improved bounds.

The current research on competitive-ratio approximation schemes focussed on particular online scheduling problems. Our vision is to use insights for particular problems to eventually characterize general properties of online problems that allow for a competitive-ratio approximation scheme.

## References

- [1] F. N. Afrati, E. Bampis, C. Chekuri, D. R. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko. Approximation schemes for minimizing average weighted completion time with release dates. In *Proceedings of the 40th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 32–43, 1999.
- [2] S. Albers. Better bounds for online scheduling. *SIAM J. Comput.*, 29(2):459–473, 1999.
- [3] Y. Bartal, A. Fiat, H. J. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. *J. Comput. Syst. Sci.*, 51(3):359–366, 1995.

- [4] Y. Bartal, H. J. Karloff, and Y. Rabani. A better lower bound for on-line scheduling. *Inf. Process. Lett.*, 50(3):113–116, 1994.
- [5] P. Berman, M. Charikar, and M. Karpinski. On-line load balancing for related machines. *Journal of Algorithms*, 35:108–121, 2000.
- [6] R. Chandrasekaran, B. Chen, G. Galambos, P. R. Narayanan, and A. van Vliet. A note on “An on-line scheduling heuristic with better worst case ratio than Graham’s list scheduling”. *SIAM J. Comput.*, 26(3):870–872, 1997.
- [7] B. Chen, A. van Vliet, and G. J. Woeginger. An optimal algorithm for preemptive on-line scheduling. *Operations Research Letters*, 18(3):127–131, 1995.
- [8] L. Chen, D. Ye, and G. Zhang. Approximating the optimal competitive ratio for an ancient online scheduling problem. Technical report, CoRR, abs/1302.3946v1, 2013.
- [9] Y. Cho and S. Sahni. Bounds for list schedules on uniform processors. *SIAM Journal on Computing*, 9(1):91–103, 1980.
- [10] T. Ebenlendr, W. Jawor, and J. Sgall. Preemptive online scheduling: Optimal algorithms for all speeds. *Algorithmica*, 53(4):504–522, 2009.
- [11] T. Ebenlendr and J. Sgall. A lower bound on deterministic online algorithms for scheduling on related machines without preemption. In *Proc. of WAOA 2011*, pages 102–108, 2012.
- [12] L. Epstein, J. Noga, S. Seiden, J. Sgall, and G. Woeginger. Randomized online scheduling on two uniform machines. *J. of Scheduling*, 4(2):71–92, 2001.
- [13] U. Faigle, W. Kern, and G. Turán. On the performance of on-line algorithms for partition problems. *Acta Cybern.*, 9(2):107–119, 1989.
- [14] R. Fleischer and M. Wahl. Online scheduling revisited. In M. Paterson, editor, *Proceedings of the 8th Annual European Symposium on Algorithms (ESA)*, volume 1879 of *Lecture Notes in Computer Science*, pages 202–210. Springer Berlin / Heidelberg, 2000.
- [15] G. Galambos and G. J. Woeginger. An on-line scheduling heuristic with better worst case ratio than graham’s list scheduling. *SIAM J. Comput.*, 22(2):349–355, 1993.
- [16] R. L. Graham. Bounds on multiprocessing anomalies. *The Bell System Technical Journal*, 45:1563–1582, 1966.
- [17] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.
- [18] E. Günther, O. Maurer, N. Megow, and A. Wiese. A new approach to online scheduling: Approximating the optimal competitive ratio. In *Proc. of SODA 2013*, pages 118–128, 2013.
- [19] D. R. Karger, S. J. Phillips, and E. Torng. A better algorithm for an ancient scheduling problem. *J. Algorithms*, 20(2):400–430, 1996.
- [20] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy paging. *Algorithmica*, 3:70–119, 1988.
- [21] A. Kurpisz, M. Mastrolilli, and G. Stamoulis. Competitive ratio approximation schemes for makespan scheduling problems. In *Proceedings of the 10th Workshop on Approximation and Online Algorithms (WAOA 2012)*, 2012. To appear.
- [22] K. R. Pruhs, J. Sgall, and E. Torng. Online scheduling. In J. Y.-T. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter 15. Chapman & Hall/CRC, 2004.



- [23] J. F. Rudin III and R. Chandrasekaran. Improved bounds for the online scheduling problem. *SIAM J. Comput.*, 32:717–735, 2003.
- [24] J. Sgall. On-line scheduling – a survey. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, volume 1442, pages 196–231. Springer, Berlin, 1998.
- [25] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985.
- [26] J. Wen and D. Du. Preemptive on-line scheduling for two uniform processors. *Operations Research Letters*, 23:113–116, 1998.